CSE 101 12-1-20

Pa6: ext. 2 days

Pa7: ext. 1 day

<u>Thm</u>

if a RBT has n (non-nil) nodes and height h, then

$$h \leq 2\lg(n+1)$$

<u>Remarks</u>

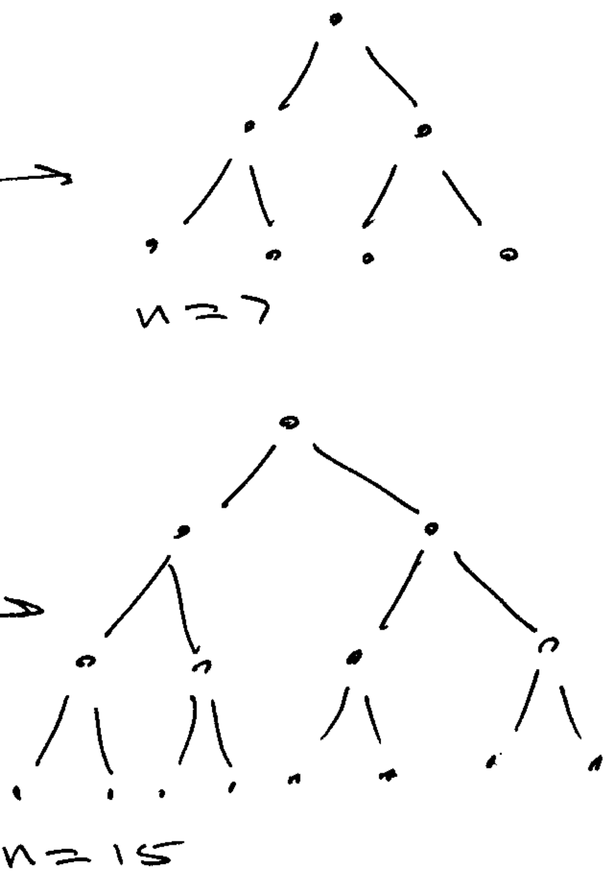
- any binary tree satisfies

$$h \geq \lfloor \lg n \rfloor$$

why ?

Complete Binary trees

CBT



# nodes	depth
1	0
2	1
4	2

# nodes	depth
1	0
2	1
4	2
8	3

If a CBT has height h, then

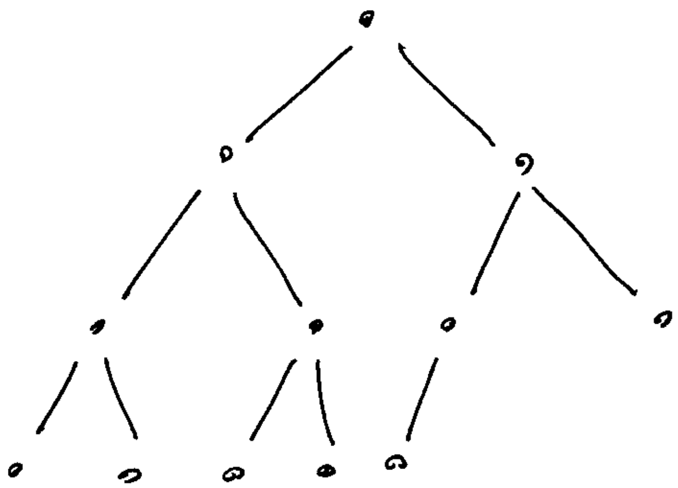
$$\# \text{ nodes} = n = \sum_{d=0}^{h} 2^d = \frac{2^{h+1} - 1}{2 - 1} = \boxed{2^{h+1} - 1}$$

$$\therefore \quad 2^{h+1} = n + 1$$

$$\therefore \quad h + 1 = \lg(n+1)$$

$$\therefore \quad h = \boxed{\lg(n+1) - 1}$$

Almost complete binary tree (ACBT)



$h = 3$
$n = 12$

let # nodes $= n$

$$2^{(h-1)+1} - 1 < n \le 2^{h+1} - 1$$

$$2^h \le n < 2^{h+1}$$

$$h \le \lg n < h+1$$

$$\therefore \boxed{h = \lfloor \lg n \rfloor}$$

so an RBT satisfies both

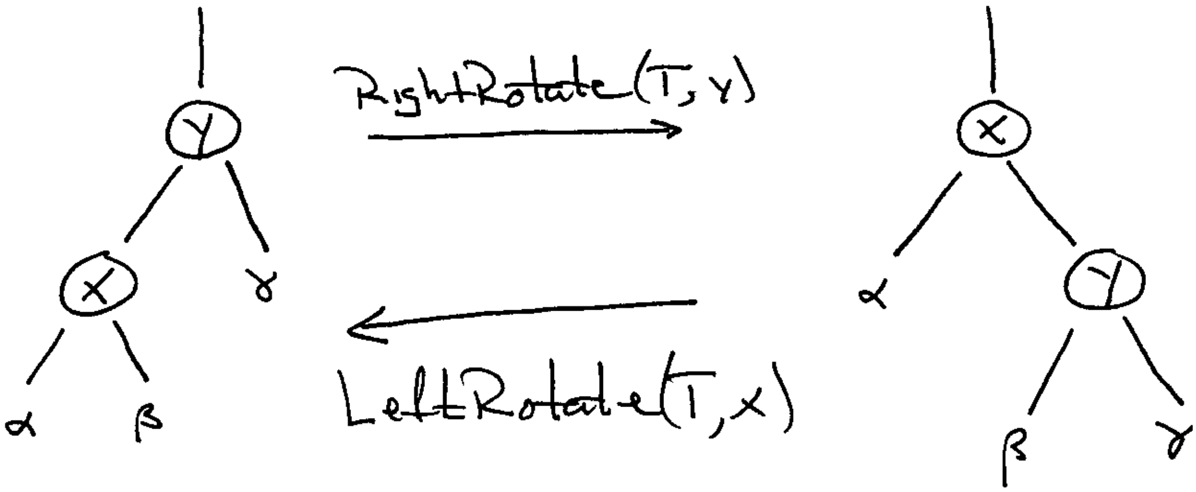
$$\underbrace{\lfloor \lg n \rfloor}_{\Theta(\log n)} \underset{\text{all BTs}}{\leq} \text{height}(T) \underset{\text{by Thm}}{\leq} \underbrace{2\lg(n+1)}_{\Theta(\log n)}$$

$\therefore$ height$(T) = \Theta(\log n)$.

- a Binary tree satisfying is called a Balanced Tree.
- Insert() and Delete( ) for BSTs do not preserve RBT properties, however they can be altered to do so and still run in time $\Theta(\text{height}(T)) = \Theta(\log n)$

13.2 Rotations

note: Insert & Delete do not preserve RBT properties.



α, β, γ are arbitrary subtrees

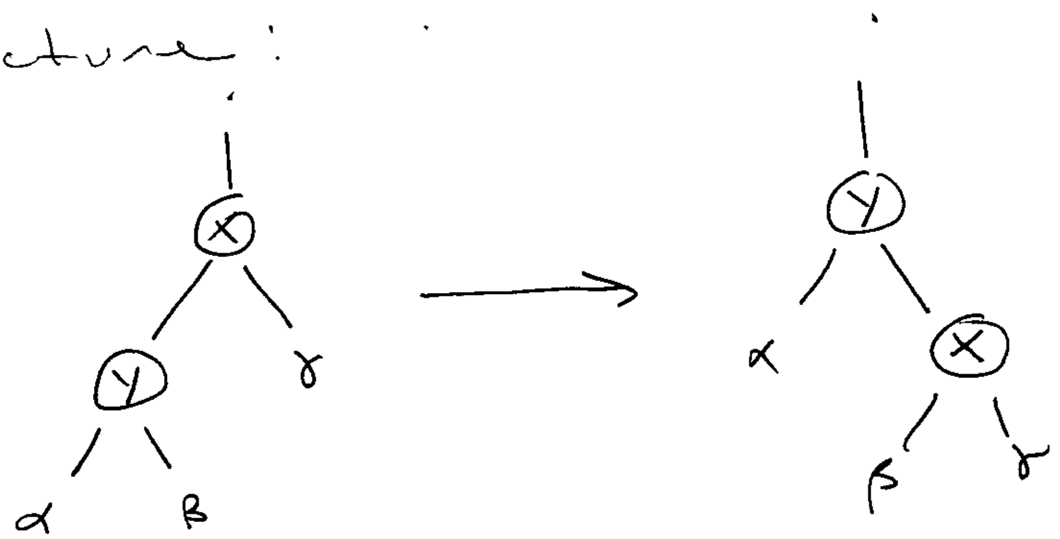
note: both operations preserve inequalities

$$key(\alpha) \le key[x] \le keys(\beta) \le key[y] \le keys(\gamma)$$

$\therefore$ both Rotations preserve BST properties.

see Pseudo-code for RightRotate(T, x)

Picture:



Summary:

- fix β: left[x] = right[y]
  p[right[y]] = x

- fix Parent: p[y] = p[x]
  p[x] left or right child = y

- fix x, y: right[y] = x
  p[x] = y

13.3 Insertion

See Pseudo-code for RB-Insert.

Question:

If we do a BST insert into a RBT, which RBT Property(s) could be violated?

1, 2, 3 : true or easy to fix

5 : can be made true by coloring new node Red.

4 : may be false.

see pseudo-code for

RBInsertFixUP(T, z)

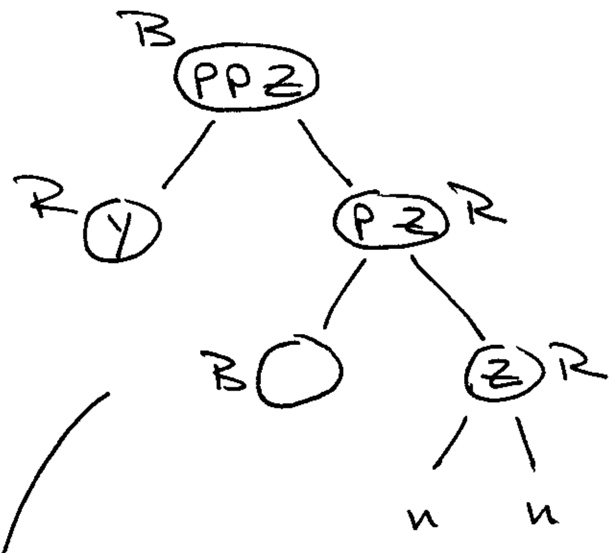
2 cases:

- p[z] is left child of its parent.
  let y be p[z]'s right sibling (z's uncle). have 3 subcases: {1, 2, 3}

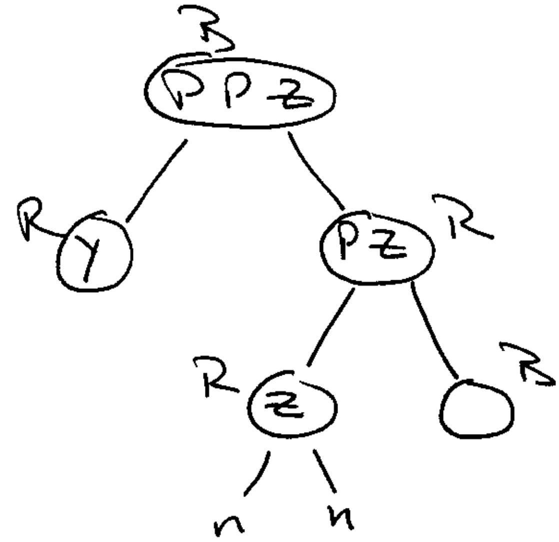
- p[z] is right child of its parent.
  let y p[z]'s left sibling (z's uncle). have 3 subcases {4, 5, 6}

we discuss cases 4, 5, 6

<u>case 4</u>: y is red.



<u>or</u>



fix color relationships between
Y, PZ, PPZ, then let z climb up
to PPZ